Students:
Nguyen Huu Truong Giang [u6110634@udg.edu]
Praveen Kumar Murali [u6110748@udg.edu]

Lab 3: A star Algorithm

1 Introduction

The A* (A-star) algorithm is an informed search algorithm that uses heuristic information to efficiently find the shortest path to a goal. It combines the actual cost from the start node to the current node with an estimated (heuristic) cost from the current node to the goal, thereby balancing exploration and exploitation during the search. A* guarantees an optimal path if the heuristic function is *admissible* (i.e., never overestimates the true cost to reach the goal). By using this approach, the algorithm avoids exploring unnecessary nodes, achieving both efficiency and accuracy in path planning.
An example of an A* path is illustrated in Figure 1.

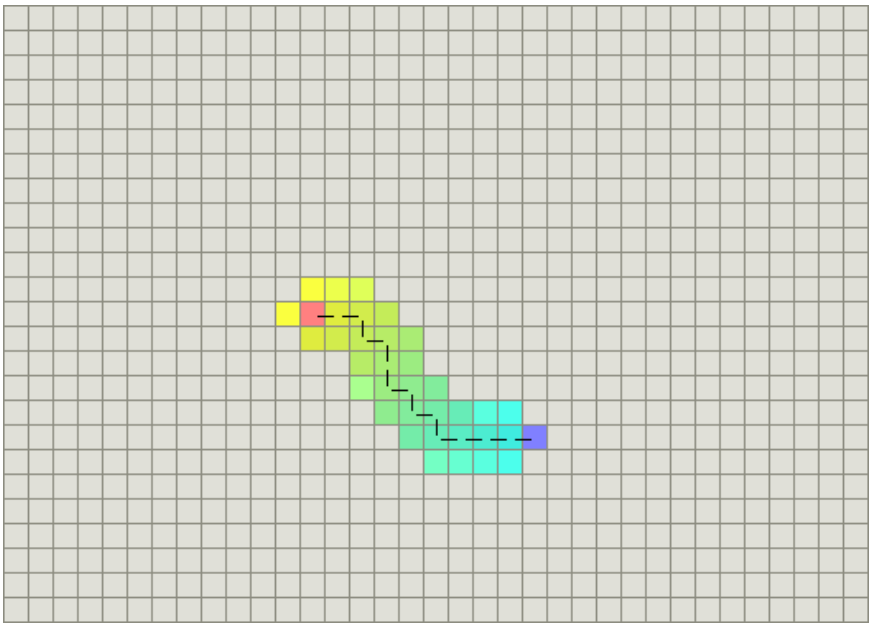

Figure 1: Visualization of A* path planning [1].

This report is organized as follows: Section 2 presents the mathematical formulation of the A* algorithm and the pseudocode used for its implementation. Section 3 discusses the performance of the algorithm on different map configurations, including an analysis of the total path cost for each case. Finally, Section 4 highlights the problems encountered during the implementation and

summarizes the main conclusions drawn from the lab experiment.

2 Methodology

This section describes the mathematical formulation and pseudocode of the A* path planning algorithm. The implementation employs the Euclidean distance as the heuristic cost function and considers two different movement configurations: *4-connectivity* and *8-connectivity*.

2.1 Euclidean Distance and Grid Connectivity

The Euclidean distance is used to measure the straight-line distance between two points p_1 and p_2 in a 2D grid map. It serves as the heuristic cost $h(n)$ in the A* algorithm, estimating the remaining distance from a given node to the goal.

$$d(p_1, p_2) = \sqrt{(p_{1x} - p_{2x})^2 + (p_{1y} - p_{2y})^2} \tag{1}$$

Where:

- $d(p_1, p_2)$: Euclidean distance between points p_1 and p_2.
- p_1, p_2: Coordinates of two points, represented as $[p_{1x}, p_{1y}]$ and $[p_{2x}, p_{2y}]$.

In a discrete 2D grid map, robot movement can be defined using different connectivity models that determine which neighboring cells are reachable from a given position:

- **4-connectivity**: Movement is allowed in four cardinal directions — up, down, left, and right.
- **8-connectivity**: Movement includes both the four cardinal and four diagonal directions, enabling more flexible path generation.

Figure 2 illustrates the difference between 4-connectivity and 8-connectivity on a grid map.

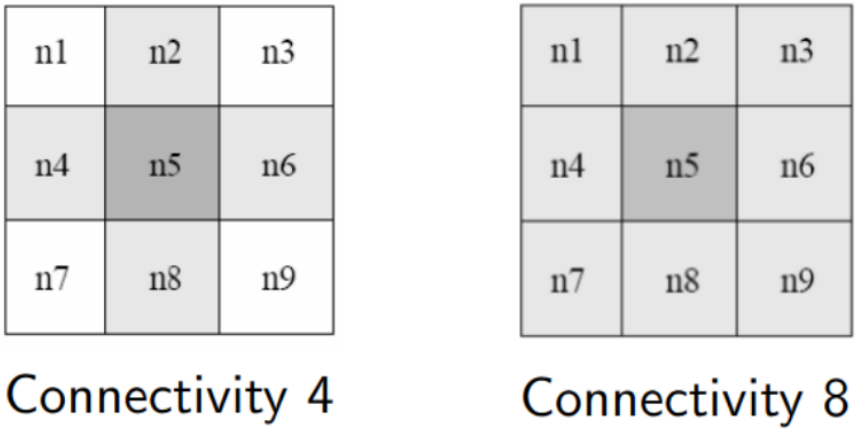

Figure 2: Illustration of 4-connectivity and 8-connectivity in a grid map.

2.2 A* Algorithm

The A* (A-star) algorithm is an informed search method that combines the actual cost from the start node with an estimated cost to the goal, known as the heuristic. By summing these two components, A* determines which node to explore next, efficiently guiding the search toward the optimal path. The total cost function is defined as:

$$f(n) = g(n) + h(n) \tag{2}$$

Where:

- $f(n)$: Total estimated cost of the path passing through node n.
- $g(n)$: Actual cost from the start node to the current node n.
- $h(n)$: Heuristic cost estimating the distance from node n to the goal.

In this implementation, the Euclidean distance is used as the heuristic function $h(n)$. The pseudocode for the A* algorithm is presented in Algorithm 1 [2].

Algorithm 1 A-star algorithm

Given heuristic $h : V \to \mathbb{R} \geq 0$, start s, goal g
Initialize $g(v) \leftarrow +\infty$, $\pi(v) \leftarrow$ null for all v; set $g(s) \leftarrow 0$
Let O be a min-priority queue keyed by $f(v) = g(v) + h(v)$; insert s into O. Set $C \leftarrow \emptyset$
while $O \neq \emptyset$ **do**
    Extract $u \in O$ with minimum $f(u)$
    if $u = g$ **then**
        return Path by following π from g to s
    end if
    u to C
    for each neighbor v of u **do**
        $t \leftarrow g(u) + w(u, v)$
        if $v \notin C$ and ($v \notin O$ or $t < g(v)$) **then**
            $g(v) \leftarrow t$, $\pi(v) \leftarrow u$, $f(v) \leftarrow g(v) + h(v)$
            if $v \notin O$ **then** insert v into O
            end if
        end if
    end for
end while
return Failure

3 Results

We conducted experiments using the **A* algorithm** on four different maps (Map 0, Map 1, Map 2, and Map 3), as illustrated in Figure 3. In each map, the **green circle** represents the start position,

while the **red star** indicates the goal position. The position configurations for each experiment are summarized in Table 1.

Map	Start	Goal
0	[10, 10]	[90, 70]
1	[60, 60]	[90, 60]
2	[8, 31]	[139, 38]
3	[50, 90]	[375, 375]

Table 1: Start and Goal configuration for each map

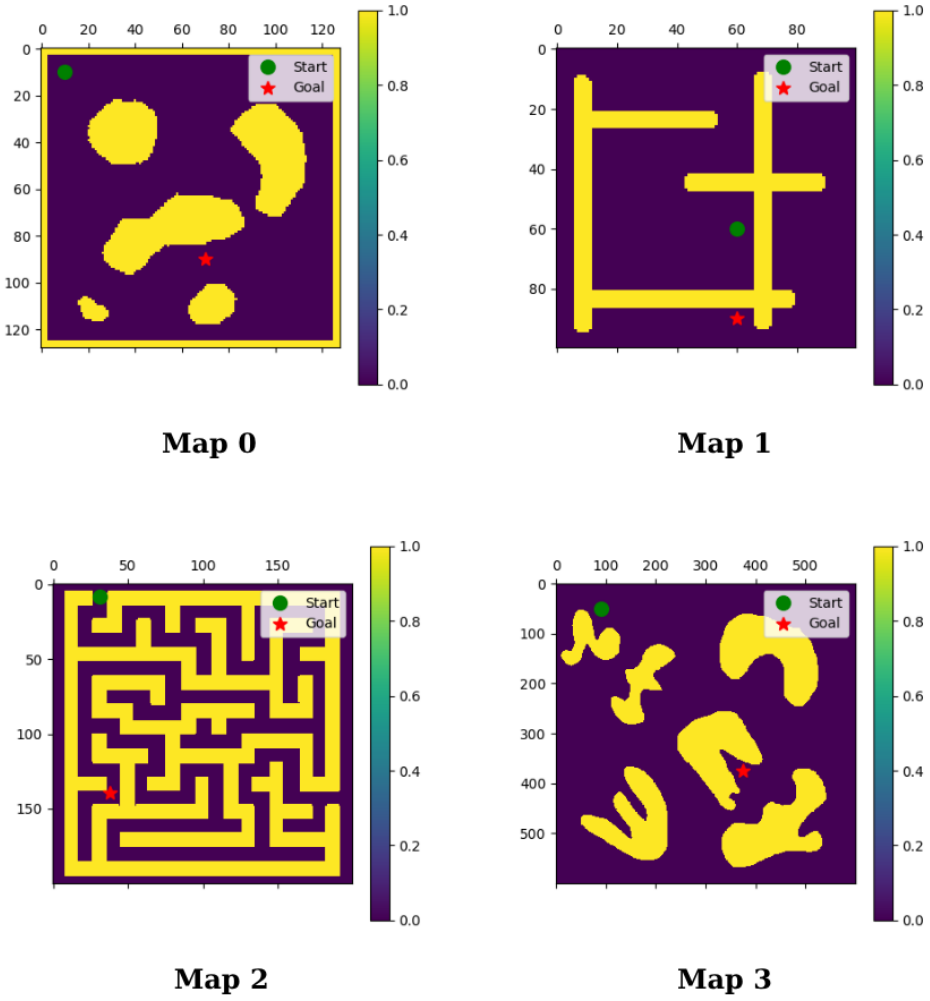

Figure 3: Map 0, 1, 2, and 3 with start and goal position

3.1 4-connectivity

When applying the **A* algorithm** on four different maps using **4-connectivity**, the results show that the algorithm successfully finds a valid path from the start position to the goal position, even in complex map environments, as illustrated in Figure 4. From Table 2, it can be observed that the *total path cost* increases proportionally with both the size and complexity of the map, demonstrating how map density and obstacle distribution affect the overall path length.

Map	Total path cost	Map	Total path cost
0	154.0	1	240.0
2	632.0	3	688.0

Table 2: Total path cost using 4-connectivity

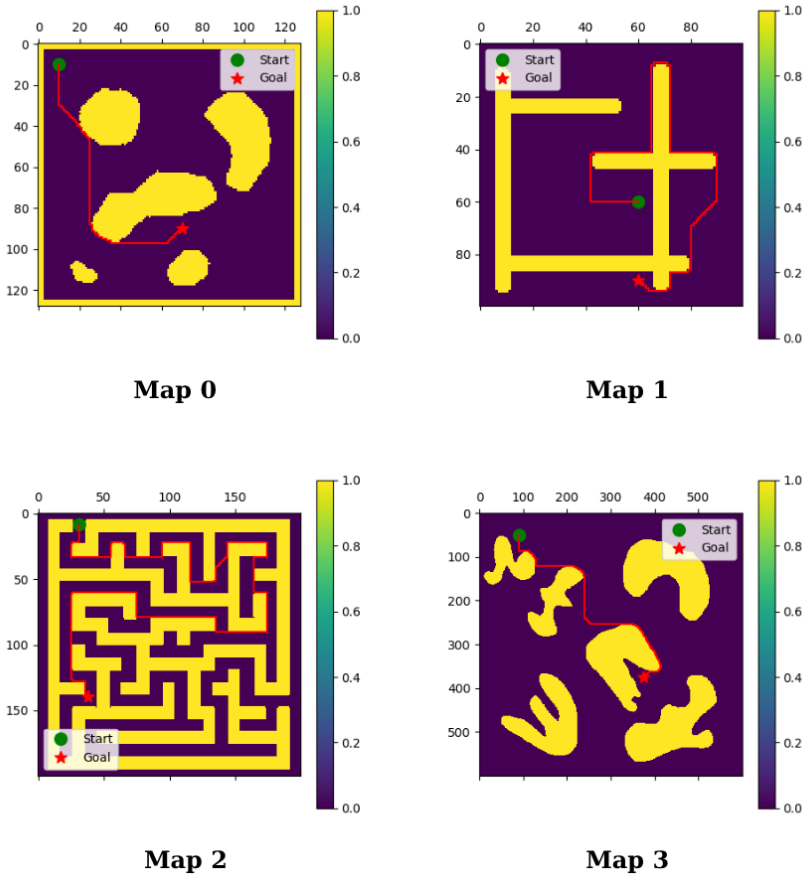

Figure 4: Path from start to goal position using A-star algorithm with 4-connectivity in different maps

3.2 8-connectivity

Similar to the 4-connectivity case, the **A* algorithm** with **8-connectivity** also successfully finds a valid path from the start position to the goal position across all four maps. However, the resulting paths differ from those obtained using 4-connectivity, as the algorithm can move diagonally, leading to more direct trajectories. As shown in Table 3, the total path cost obtained with 8-connectivity is consistently lower than with 4-connectivity, demonstrating the efficiency gained by allowing diagonal movements. The resulting paths are illustrated in Figure 5.

Map	Total path cost	Map	Total path cost
0	133.58	1	191.97
2	555.85	3	523.39

Table 3: Total path cost using 8-connectivity

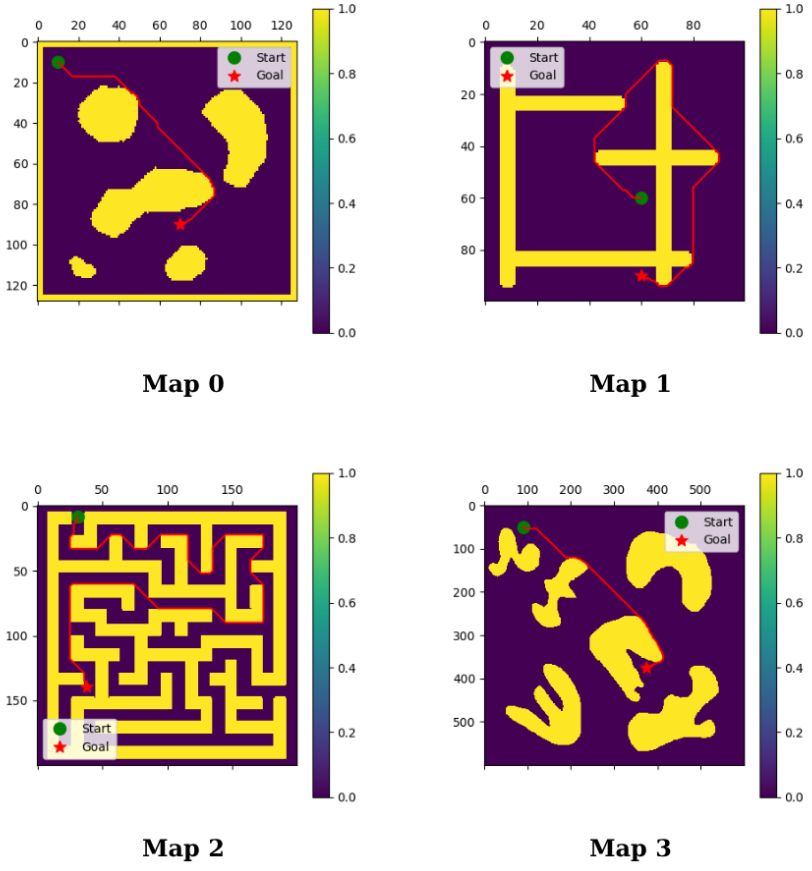

Figure 5: Path from start to goal position using A-star algorithm with 8-connectivity in different maps

4 Discussions and Conclusions

The **A* algorithm** is an intuitive and widely used path planning method that can be implemented straightforwardly in code. However, during implementation, several common issues may arise. One frequent mistake is neglecting to use a *min-priority queue* (often implemented with a heap), which is essential for efficiently selecting the next node with the lowest total cost $f(n)$. Without it, the algorithm may not function correctly and could fail to find a valid path.

Another critical consideration is the choice of heuristic function. If the heuristic cost $h(n)$ is *overestimated*, the algorithm may no longer guarantee an optimal solution. To verify the admissibility of the heuristic, one can initially set $h(n) = 0$, which effectively transforms A* into Dijkstra's algorithm. Since Dijkstra's algorithm always finds the optimal path, comparing its result with the A* solution allows us to confirm whether the chosen heuristic maintains optimality.

In summary, the **A* algorithm** is a powerful and efficient pathfinding approach that avoids local minima and minimizes unnecessary exploration by combining actual and heuristic costs. When implemented correctly with an admissible heuristic and an efficient data structure, it achieves optimal and computationally efficient path planning.

References

[1] A. Patel, "Introduction to a*," `https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html`.

[2] N. Palomeras, "Slide autonomous systems graph search," *Universitat de Girona*. `https://moodle.udg.edu/pluginfile.php/2491457/mod_resource/content/3/graph_search.pdf`.